THE PRIVILEGE CONTROL TABLE TOOLKIT:
AN IMPLEMENTATION OF THE SYSTEM BUILD APPROACH

Thomas R. Woodall
Hughes Aircraft Company
2000 E. Imperial Highway
El Segundo, CA 90245
310-334-7603
FAX: 310-334-1242
twoodall@msmail4.hac.com

Roberta Gotfried
Hughes Aircraft Company
2000 E. Imperial Highway
El Segundo, CA 90245
310-334-7655
FAX: 310-334-1242
rgotfried@msmail4.hac.com

1. Abstract

This paper describes the Privilege Control Table (PCT) Toolkit as developed for military real-time embedded avionics systems. This tool is the evolutionary result of research and development of the trusted 'System Build' concept as originally described in [SYSBUILD] and refined in [SYSBLL]. This paper describes what the tool does and how it is used.

Keywords: System Build, Real-time, Multilevel Security

2. Introduction

The shrinking DoD budget is driving military weapon system architects to design affordable systems with greater lethality, survivability, and flexibility requirements. This is particularly true of air vehicle weapon systems, where funding limits the number of systems and aircraft that can be developed and deployed. For avionics systems, this is one of many factors that has led to integrated avionics architectures. Integrated avionics systems cost less than comparable federated systems and have the advantage of being able to provide information with increased accuracy and understandability to the pilot. To further enhance lethality, real-time sharing of information among coordinating assets is now required. In effect, the avionics system of one aircraft is one node of a distributed C4I network. This leads to an avionics system that is processing an extremely wide range of information, from unclassified weather data to highly classified and compartmentalized information from off-board assets. This results in the need for a reliable, high performance, multilevel secure, embedded avionics system.

In past avionics systems, the hard scheduling deadlines provided difficult design challenges. Now, these real-time systems must additionally meet stringent security requirements. In order to mitigate the security impact on real-time performance, the 'System Build' concept was developed as described in [SYSBUILD] and [SYSBLL].

This paper focuses on the PCT Toolkit, a System Build tool. Section 3 briefly summarizes the System Build approach and its motivations and enabling factors; it also gives a system overview of the PCT Toolkit. Section 4 describes in detail what functions the PCT Toolkit performs, Section 5 discusses outstanding security issues associated with the Toolkit, and Section 6 summarizes the advantages and disadvantages of the PCT Toolkit and identifies areas of further work.

3. The System Build Approach

The System Build approach was developed as a result of merging two seemingly conflicting requirements: hard real-time performance and multilevel security. Few systems have had to meet both requirements simultaneously, particularly in an embedded avionics environment. Results from research in real-time and secure systems provided inadequate solutions to meeting requirements for systems to be built (target systems).

3.1 System Characteristics

Integrated avionics systems comprise a closely connected network of heterogeneous processing nodes connected to a set of sensors. Real-time characteristics of these systems are described by specific performance requirements in terms of throughput, I/O and data bandwidth, as well as interrupt latency constraints in microseconds. Such systems use preemptive priority-based process scheduling.

Mission success depends on the ability of the system to process sensor data within time constraints as well as to adapt rapidly to changes in the environment.

It is essential that security mechanisms not adversely impact the ability of the system to meet timelines, or respond to external events in a timely manner. Even small impacts in response times can have an adverse affect on the ability of the system to perform functions such as detecting and tracking targets and responding to events. Security must be achieved without sacrificing real-time performance.

3.2 Subjects/Objects Known *A Priori*

A key enabler of the System Build approach is that all subjects and objects are known *a priori*. This is a result of the embedded nature of the avionics environment. The *a priori* method is also applicable to a wide range of systems that have a fixed set of application configurations over a period of time.

The System Build approach allows access mediation at the time the software configuration is "built." All interactions of security subjects and objects are known and may be verified for consistency with the system security policy before the software is loaded into the target system. The build-time tool that performs this function is the PCT Toolkit.

3.3 Built for a Specific Configuration

The set of system subjects and objects for a specific software load is given a unique identifier and is known as a System Build Configuration. In today's avionics systems, a software load may not change for several months. Furthermore, the hardware configuration associated with the software will also not change for the lifetime of the software load. This fixed combination of hardware and software components is also called the System Build Configuration. See Figure 3-1.

The inputs to the PCT Toolkit are a subset of the information that defines a System Build Configuration. Other information includes system attributes, such as the hardware configuration, that are held constant for all capabilities.

The definition of a System Build Configuration allows for a range of System Build Capabilities to be supported. For example, a System Build Capability might define the possible combinations of sensor data processing for a specific set of available processing assets. The specific set of capabilities used in a system load depends on factors which are transparent to the Toolkit. However, each specific capability must be defined by the system designer so that the Toolkit can determine what set of subjects may be allowed to run simultaneously.

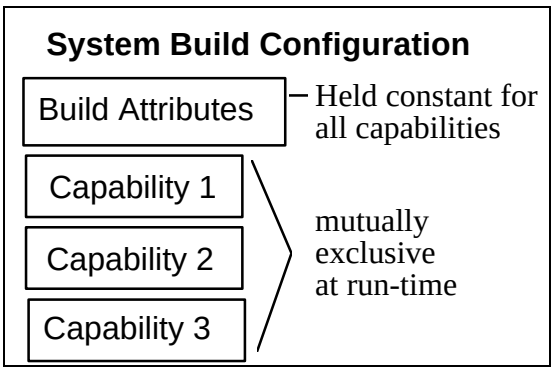

Figure 3-1: A System Build Configuration

3.4 System Engineering Methods

Embedded avionics systems are developed using rigorous engineering methods. System complexity and size, in conjunction with safety and mission criticality factors, have led to a methodology that results in extensive design analysis and documentation. Tool support for system and software engineering methods results in on-line "databases" and "documentation" that can be leveraged in implementing System Build. Application/software/hardware interfaces are specified and reviewed as a matter of course. The PCT Toolkit benefits from these methods by using the interface definitions, along with the system security model to validate all interactions.

Results of the Toolkit analysis can be organized into tables that can be used efficiently at run-time. Each table, called a privilege control table, contains only the validated interactions and resources for a particular security subject. The operating system [AOS] performs enforcement at run-time. Implementation of the AOS and PCT is discussed further in Section 4.

Invalid interactions are logged for action by a system administrator (or designated integration team personnel during system integration).

3.5 TCB View With System Build

The System Build approach results in a system that minimizes the real-time performance of multilevel security measures. The side effect is that the trusted computing base (TCB) has been distributed and possibly enlarged. Figure 3-2 shows how the TCB is distributed in the System Build approach.

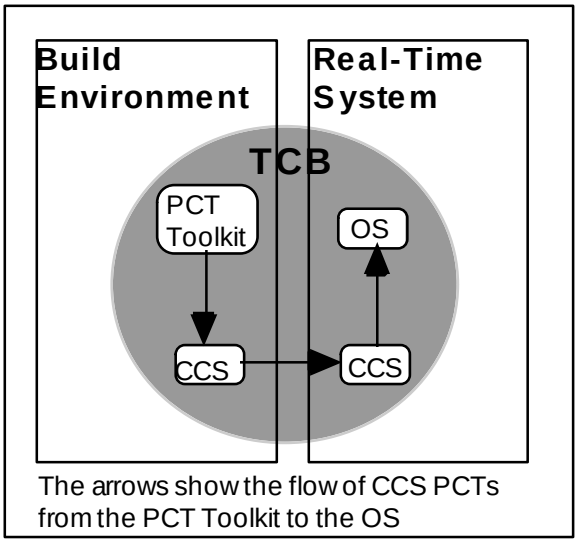

Figure 3-2: The System Build Approach

The separation and distribution of the TCB raises several issues, some of which are raised in [SYSBLL].

One issue is the effect of System Build on certification and accreditation (C&A). C&A of the TCB must be performed on the real-time operating system in concert with the Toolkit. Lack of precedent for this approach impacts the first such system, but should be ameliorated in future instantiations.

Trusted distribution of PCTs from the build environment, a secure contractor, or government site, to the real-time environment in the embedded system also must be addressed. This problem is solved today by use of a cryptochecksum (CCS), a capability already available in the target system.

3.6 How PCT Toolkit fits into the Software Development Process

Figure 3-3 shows the integration of the PCT Toolkit into the Software Development process. PCT development basically parallels software development. In fact, at each stage the products must be consistent. That is, the source code and the interface information must be consistent or the resulting PCTs and executable software will not be consistent. Inconsistencies can result in a variety of failures. Section 5.3 addresses these consistency issues in detail.

At run-time, when the OS loads a program, it also loads its corresponding PCT. The PCT can only be accessed by the OS. The combination of a software program together with a PCT defines a subject.

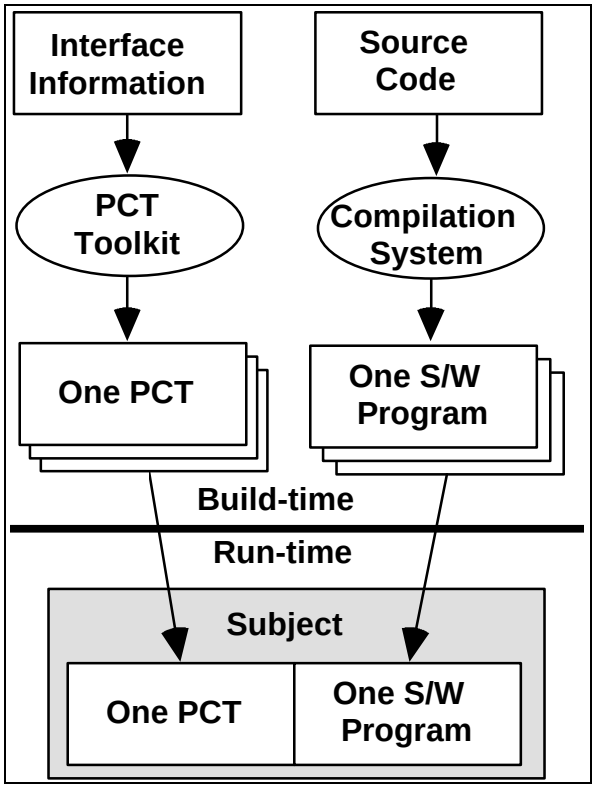

Figure 3-3: The PCT and Code Development Processes Through Run-Time

4. The PCT Toolkit in Detail

The fundamental purpose of the PCT Toolkit is to perform mediation on an information flow model of the system during System Build and to encapsulate the relevant information and privileges into tables that can be used at run-time by the OS, thereby reducing the processing assets dedicated to security in the real-time environment. Figure 4-1 shows the Toolkit data flow.

In order for the PCT Toolkit to accomplish this task it must do the following:

1) Understand the Target Environment

2) Build an Information Flow Model from Interface Definitions
3) Apply the Target Environment Information to Information Flow Model (a.k.a. Consistency Checks)
4) Perform mediation using the Bell and LaPadula [BLP] model on events requiring mediation
5) Output PCTs and other tables
6) Output reports to assist users
7) Generate a log file to document all input information, all conclusions (e.g., security violations), and all outputs generated.

In addition to the above, the PCT Toolkit allocates resources within some object classes. As will be shown, this is consistent with its purpose.

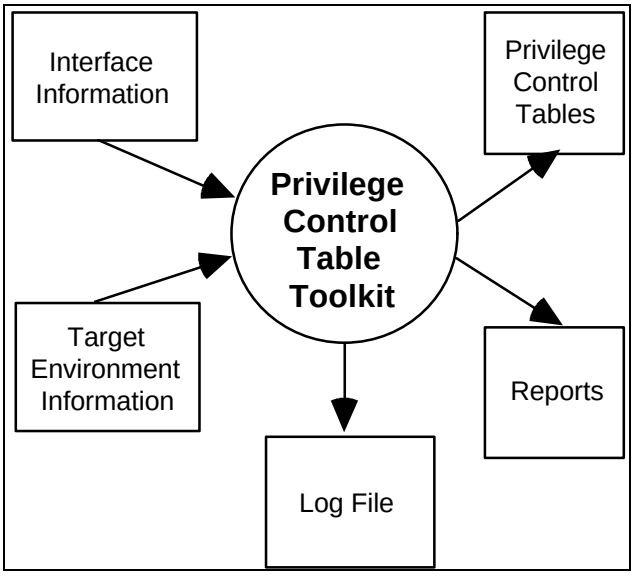

Figure 4-1: PCT Toolkit Data Flow

4.1 Understanding the Target

In order for the PCT Toolkit to accomplish its purpose, it must have knowledge of the target environment. Since the target environment may change considerably from one system to another, the Toolkit must be flexible; therefore, at initialization the PCT Toolkit reads in information that configures the PCT Toolkit for the target environment. This information is contained in configurable knowledge (CK) files. The CK files allow for definition of the hardware implementation, of the security classifications, and of privileged OS services.

4.1.1 Hardware Information

The hardware CK files describe the hardware components and their quantity, interconnect topology, and relevant security capabilities. For example, security characteristics of special purpose processors may be described by the number of allowable programs served, and the ability of the processor to maintain separation of data.

4.1.2 Security Information

One CK file read by the Toolkit defines all classifications in the system and their dominance relationships. The Toolkit has no built-in knowledge about classifications. The security CK files completely defines the semantics of classifications. It supports categories and allows for disjoint classifications to be defined. In addition, as shown in Figure 4-2, it can be defined such that two or more classifications can actually be made equivalent for the purposes of applying the security policy.

4.1.3 OS Information

The PCT Toolkit and the OS are closely tied. As in most OSs, there are some privileged services that are very limited in their use. The subjects that are allowed to use these services are specified in a CK file on a per service basis.

-- Classification Set Definitions

```
--**********************************
classification_set    : LevelSet1
classification        : Unclassified
classification_set    : LevelSet2
classification        : Confidential
classification_set    : LevelSet3
classification        : Secret
classification_set    : LevelSet4
classification        : Secret/SAR/CAT1
classification        : Confidential/SAR/CAT1
classification_set    : LevelSet5
classification        : Secret/SAR/CAT2
classification        : Confidential/SAR/CAT2
classification_set    : System-High
classification        : System High

-- Dominance Relation assertions
-- (transitivity holds)
--**********************************
assert  : System-High dominates LevelSet4
assert  : System-High dominates LevelSet5
assert  : LevelSet5 dominates LevelSet3
assert  : LevelSet4 dominates LevelSet3
assert  : LevelSet3 dominates LevelSet2
assert  : LevelSet2 dominates LevelSet1
```

Figure 4-2: Security CK File

4.2 Building an Information Flow Model

After the Toolkit has processed the Target Environment, it can build an information flow model based on interface information defined prior to running the PCT Toolkit. Two methods of gathering interface information have been implemented. The first involves querying a database based on user input. The second involves reading a set of user-specified interface definition files (IDFs).

The database method is driven by a user specifying a set of System Build Capability Tables. There is one table for each capability, each table listing the subjects that must exist for that capability. Based on that list, the Toolkit will query the database for necessary subject and object information.

The IDF method utilizes one IDF per subject, and the IDF contains all object access requests. An IDF is divided into two parts. One defines the subject and the other defines object access requests. In the object section there are subsections for each object type. At least one IDF in the system must identify the object attributes for a given object.

Hereafter, the term IDF shall be used to mean either method. While reading in the information, the PCT Toolkit performs semantic checks, e.g., verifying that every classification used in an IDF is defined in the Security CK file.

4.3 Consistency Checks

The PCT Toolkit checks data consistency to ensure input data are well defined. For example, one consistency check is for readers and writers of volatile objects, that is, objects that exist only when power is present. The PCT Toolkit checks to be sure that there are both readers and writers of every such object. If not, a warning message will be generated. In addition to general data consistency checks, security-relevant consistency checks are also performed by the Toolkit.

The PCT Toolkit will verify that all subject and object names are unique. For subjects, this means verifying that there is only one subject definition. If the Toolkit reads two subject definitions using the same name it will generate an error. For objects, the Toolkit assumes there will be only one definition: if it encounters attributes about an object with the same name, it will check that any attributes specified more than once are identical. Error messages are generated for all inconsistencies.

Some object classes allow only one writer per object. The Toolkit will check that this rule is followed per System Build Capability.

Another consistency check deals with verifying that software is correctly mapped to hardware resources. One or more of the attributes of a subject relate to the hardware needs of a subject. The Toolkit will verify that all such mappings are compatible.

4.4 Supports Resource Allocation

As the PCT Toolkit concept began to evolve from the System Build approach, several potential enhancements became apparent. One such enhancement was global resource allocation, that is, allocation of resources shared by subjects that are not on the same processor. The Toolkit will bind names to specific physical resources.

For example, the Toolkit knows the attributes of all interprogram messages including all senders and receivers, and also has the software/hardware mapping. Therefore, the Toolkit has all the information necessary to assign bus labels, and its place and role in development process makes it ideal to accomplish this task.

Global resource allocation results in a single PCT object entry that contains the rights of the subject to access the object, and also a mapping of the object to its low-level resource identifier, e.g., a bus label. Because this information is created at build-time and is distributed in a trusted manner via the PCTs, no run-time consumption of resources is necessary to accomplish this task. This enhancement of the System Build approach is consistent with PCT Toolkit's purpose, which is to make security affordable, from a timing perspective, in the system.

The PCT Toolkit supports several types of global resource allocation, although system requirements will dictate which are possible for a specific Toolkit implementation.

4.5 Build-Time Mediation

The Security Policy Model applied by the PCT Toolkit is based on the Bell and LaPadula model [BLP], although another model could be substituted into the tool.

4.5.1 Applying IDF Classification Labels

The interface information contained in the IDFs is labeled according to the actual classification of the information, hereinafter called content labeling. The PCT Toolkit labels information according to its container, hereinafter called container labeling.

Therefore, if an untrusted HIGH subject sends a message, M, labeled in an IDF as LOW, the PCT Toolkit will classify the message M as HIGH. A read request of M in an IDF by a HIGH subject will therefore be allowed. A read request of M in an IDF by a LOW subject will cause a security error; however, the error will be traced back to the offender according to content labeling. Therefore, the error message will state that there was an illegal attempt by a HIGH subject to write a LOW message.

This method is consistent with the model, and yet preserves the user view of the system as described in the IDFs. Furthermore, no information in the IDFs must be artificially overclassified so that the mediation will pass.

The side effect (which is correct) is that information moving in the system outside of the domain of the PCT Toolkit and its corresponding OS must be protected at the level as classified by the Toolkit.

There are several object classes in the system and the security rules applied to a class may be unique. For example, write access at run-time to some object classes does not return status (write up allowed), whereas for others there is status returned (write up not allowed to minimize covert channels).

4.5.2 Hardware-Specific Checks

Different processors and hardware engines, e.g., a graphics display engine, have different capabilities with respect to security. The PCT Toolkit will use the definitions in the CK file to determine what additional checks must be applied. The Toolkit is not coded for specific checks; rather it determines the correct way to model the hardware and applies the applicable general rule. For example, if the hardware cannot keep different users separate and if it has both read and write capabilities, then the PCT will model the hardware as an object that inherits the classification of its user(s) and to which each user has read/write access.

4.6 Build Tables

Once the PCT Toolkit has built a system model and applied the security policy, it can create the tables necessary for the run-time TCB components to establish a secure state and to enforce the mediation decisions made by the Toolkit.

These tables include not only PCTs but also other tables. This is another case where the PCT Toolkit has taken on greater capabilities than those defined in the original System Build approach.

4.6.1 System Start-Up Information

One output is the System Start-Up Table (SST). At the start-up of the target system, the TCB software that controls the master nonvolatile memory comes up, reads and uses the SST, and then distributes the SST in a secure and reliable manner to each of the distributed start-up programs in the TCB. These programs use the SST to control local start-up. The SST indicates what TCB components should be loaded and the physical resources that should be used during the start-up process. Since the start-up scenario can change based on the System Build Configuration, and since the PCT Toolkit allocates resources and knows the size of objects, the Toolkit can create the SST that contains information on what is to be loaded, how big it is, and the resources that should be used.

4.6.2 System Manager Tables

The PCT Toolkit also builds a set of tables that are used by System Manager, a high-level program that controls the system, including what System Build Capability is used. System Manager is part of the TCB.

Because the Toolkit knows all programs needed to accomplish a particular capability in the System Build Configuration and knows what hardware is needed, it can build tables to assist the System Manager.

Immediately after start-up, the System Manager program will assess the availability of the hardware assets and based on that will determine which capabilities can be met from the set defined in System Build Configuration. It then chooses the best capability based on predetermined input and/or pilot input. Then, using tables built by the Toolkit, the System Manager can determine what programs and what PCTs need to be loaded. It is the System Manager that communicates with the OS to coordinate loading of a program and its PCT.

4.6.3 PCTs

There is one PCT created for each subject in the target system.

The internal organization of a PCT is very straightforward: there is a header plus one section per object class. The header contains a timestamp, the System Build Configuration identifier, subject information, and other information about the creation of the PCT.

Each object class section is ordered lexicographically by object name, only for those objects accessed by that subject; hence only non-null entries are present in the PCT. The PCT does not contain the object names. In order to save space and to effect an efficient run-time look-up into the PCT, the user program when calling the OS will specify an object by using its relative lexical order for a given object class. Hence, if the object name is first when ordered with the other objects of that class which are accessed by that subject, then “1” would be specified to the OS.

Unique subject and object identification is contained within each PCT. Each PCT header contains the unique subject identifier. Each object can be uniquely identified by its object class and resource identifier. The resource identifier is either the physical identifier assigned by the PCT Toolkit or it is a system-unique identifier for the class used to allocate run-time resources.

Many object classes allow for frequency of access to be specified, and this information is contained in their object entries. For these

classes the OS can perform denial of service. There may also be priority information in each entry for some object classes.

4.7 The Log File and Reports

The PCT Toolkit creates a log file as it executes, which documents all Toolkit operations relevant to a reviewer. This file is always generated and includes:

- A timestamp and other information relevant to configuration management
- A list of all user inputs and options
- All input files read
- All output files generated
- All security violations
- All other errors and warnings

The Toolkit also creates several reports that allow the user to see the information from various perspectives. One report details the information contained in each PCT. There are also reports that organize information by System Build Capabilities. Other reports detail information on all objects or specific object classes. There is also an ASCII table file that organizes all information related to the system information flow model created by the Toolkit. In this way others can build their own tool to read and organize the information that best suits their specific needs.

The Object Cross Reference report is of particular significance. There is one report per System Build Capability. It contains all objects for that capability, together with their object identifiers and classifications. The classification from the IDF information (content labeling) is listed with the classification at which the information must be protected (container labeling).

5. PCT Toolkit Security Issues

5.1 Review Issues

The PCT Toolkit works under the assumption that the information it uses is complete and correct and that it has been reviewed and approved by appropriate personnel. Therefore, if interface information states that a subject needs a specific type of access to an object, the Toolkit will assume that the access request is consistent with the principle of least privilege.

It should be noted that some CK files are more security relevant than others; therefore, the PCT Toolkit uses several CK files so that the review of the Toolkit for certification and accreditation (C&A) is simplified.

CK files were chosen over IDFs for specifying privileged OS services so that this very sensitive information would be located in one small file for easy review. Although there are generally only a handful of such services, they generally give very powerful privileges. Therefore, ease of review was a primary concern. These services are also very OS dependent. This use of CK files keeps the more general IDF information from becoming OS specific.

5.2 Comparison of Input Methods

The database and IDF input methods have different advantages and disadvantages. The advantage of the database method is that the information is logically grouped and there is no repetition that can lead to inconsistencies. Its main disadvantage is that the information used by the Toolkit is not readily reviewable: the reviewer must correctly use database tools to review it, and the review procedures are not well defined. Compounding the review problem is the fact that this method is likely to contain a lot of information that is not of immediate interest to the reviewer. Another disadvantage of this method is that the Toolkit is closely coupled to the database.

The use of IDFs is seen as the better implementation from a security review standpoint: IDFs are easy to review and correspond closely to what is contained in a PCT. However, IDFs require the PCT Toolkit to perform more extensive data consistency checks than the database approach. For example, all IDF references to object O will be checked to see that all IDFs that define O attributes define them to be the same. One such attribute is the security classification of the object.

5.3 PCT Entry Ordering

The lexigraphical ordering of PCT entries within an object class may at first seem rather burdensome since the user must know this order when calling the OS, but it is in fact quite simple to implement using high-level programming languages. For example, in the Ada programming language, the user only need create an enumerated type for each object class. The only burden upon the user is to alphabetize the symbolic names in the enumerated type. One user generates these enumerated type definitions in Ada packages using a tool which accesses the database.

This ordering method has several benefits. It is efficient from both a space and size perspective. It also allows the user to use the same symbolic names as defined in the database. Finally, a side benefit is that if good programming practices are used, the code becomes rather easy to review in terms of the objects to which a program makes reference.

This ordering implementation does not introduce any security problems. The IDF information defines what objects a subject has access to, assuming the mediation checks pass. Hence, if the IDF defines access to five objects of a given class, then the PCT section will contain five entries, with the appropriate access rights. Since all untrusted subjects operate at a single level, any erroneous access within the allowed range will be just that - a programming error. The PCT only contains approved accesses, hence the subject cannot access anything it does not have the rights to access. However, if the subject is trusted and has access to objects of differing classifications, then a potential problem exists; however, trusted programs should be closely reviewed for correctness and any reviewer instructions would explicitly state this as an item for careful review.

6. Conclusion

6.1 Advantages and Disadvantages Summarized

Implementation of the PCT Toolkit has revealed the following advantages of the System Build approach:

• The approach allows us to meet exacting system performance and security requirements simultaneously.
• The approach fits in well with the system and software engineering process, ensuring integrated system security engineering.
• System integration time is reduced due to detection of design and coding errors at system build time. This also reduces life cycle cost by decreasing the number of hours needed for integration in the target environment.
• Run-time reliability is improved as a result of reliability requirements in support of security, and as a result of finding security errors prior to run-time.
• System start-up and initialization is simplified, allowing for faster turnaround and takeoff.

The following drawbacks to the approach have also been identified:

• The approach increases the Trusted Computing Base in size and span.
• It takes longer to perform system updates. Since updates occur infrequently in the current environment, this does not have a significant impact today, but would have to be addressed if the situation changes.
• Changes to subjects/objects and the addition of new subjects requires a system-wide update of the System Build.
• Output of the PCT Toolkit is dependent on the quality and reliability of input data. This is handled today through tool support and development methods (e.g., code reviews).

These drawbacks have been addressed for the current implementation, but to achieve generalization of the approach it is desirable to mitigate these with standard methods.

6.2 Work To Be Done

We believe the viability and benefits of the System Build approach have been demonstrated in its implementation in the form of the PCT Toolkit. There is additional work that should be done to improve on the current methods, mitigate some of the drawbacks of the current instantiation, and generalize the approach to make it more widely applicable.

The PCT Toolkit was designed to allow for the application of various security policies and models. We believe it would be worthwhile to test this by applying new models (e.g., Rushby Non-Interference) and studying their impact on the tool.

We anticipate that there will be times when a new subject or object will be added to a system. An analysis should be done to determine the viability of adapting System Build and PCT Toolkit for this scenario.

The System Build approach integrates system security engineering with software and system engineering. By extending the integration of the methods and tools, we believe several results can be achieved: increased reliability of input data, improved portability to other software engineering environments, and decreased time to generate a System Build Configuration.

7. References

[AOS] M.M. Bernstein and C.S. Kim, "AOS: An Avionics Operating System for Multilevel Secure Real-Time Environments," Computer Systems Application Conference, December 1994.

[BLP] D. E. Bell and L.J. LaPadula, "Secure Computer Systems: Unified Exposition and Multics Interpretation," MTR-2997, The MITRE Corp., Bedford, MA, March 1976.

[SYSBUILD] J.P. Alstad, T.C. Vickers Benzel, et al., “The Role of System Build in Trusted Embedded Systems,” Proceedings of the 13th National Computer Security Conference, Volume 1, October 1990.

[SYSBLDLL] T.C. Vickers Benzel, M.M. Bernstein, et al., “Real-Time Trust With ‘System Build’: Lessons Learned,” IEEE, 1993.